MSX

truuks en tips

deel 8

Marcel Kreeft



MSX

truuks en tips
deel 8

Marcel Kreeft

uitgeverij STARK-TEXEL b.v.
postbus 302 1794 ZG Oosterend tel. 02223 - 661

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

MSX

MSX truuks en tips. – Oosterend:Stark-Texel
Deel 8 / Marcel Kreeft
ISBN 90 6398 850 8
SISO 365.3 UDC 681.06 NUGI 434
Trefw.: programmeren (computer) / MSX (computer)

1e druk 1987
ISBN 90 6398 850 8

Inhoud

Inleiding

Hoe meer 'Truuks en Tips' deeltjes er verschijnen, hoe moeilijker het wordt om met nieuwe, originele routines te komen.
Het blijkt echter dat MSX zo flexibel en doordacht in elkaar zit, dat het nog wel een paar jaartjes zal duren voordat iemand hem helemaal van top tot teen kent. Dit 'Truuks en Tips' deeltje is daar een voorbeeld van. Ik heb geprobeerd om een zo gevarieerd mogelijk menu aan te bieden, zodat er voor elk wat wils is. Tevens heb ik een tweetal wat langere routines opgenomen waar ik velen een plezier mee doe, denk ik. De eerste is een "Smooth-scroll" programma, dat teksten vloeiend over het scherm laat scrollen. De tweede is naar mijn mening de topper in dit boekje. Het is een utility waarmee u meer dan 80 tekens per regel op uw scherm kunt krijgen.

Al met al wens ik u veel MSX-plezier met de routines in dit boek.

Januari 1987,
Marcel Kreeft

Anti-scroll

Soms is het nodig om op de onderste regel van het scherm een tekst te printen. Als u dan de allerlaatste positie van die regel gebruikt, krijgt u onherroepelijk een omhoog-scrollend beeld. Hier helpt geen puntkomma of locate meer. Hoe kunnen we nu toch op die laatste regel iets printen? Het antwoord staat op adres &HF3B1. Op dit adres staat namelijk het aantal regels op het scherm. Normaal is de inhoud van dit adres 24. Het scherm telt immers 24 regels. Dit is tevens het maximale aantal regels dat we op het scherm kunnen zien. Zo gauw als we voorbij die regel 24 iets printen, scrollt het scherm.

We kunnen de computer echter foppen en laten denken dat regel 24 niet de laatste regel is, maar dat er 25 regels op het scherm staan. Dit doen we door op locatie &HF3B1 de waarde 25 te POKEn. Nu kunt u wel op regel 24 printen, want de computer beschouwt regel 25 pas als de laatste regel.

Onderstaand programma verduidelijkt dit nogmaals.

```
100 '============================
120 ' TEKST OP DE ONDERSTE REGEL
130 '   PRINTEN ZONDER DAT HET
140 '       SCHERM SCROLLT
150 '============================
160 ' &HF3B1 => `24' WEL SCROLL
170 '           `25' GEEN SCROLL
180 '============================
190 :
200 SCROLL=&HF3B1
210 :
220 SCREEN 0:WIDTH 39:KEY OFF
230 :
240 LOCATE 0,19
250 PRINT"NORMAAL GESPROKEN SCROLLT"
260 PRINT"DEZE TEKST NAAR BOVEN."
265 PRINT"ZOALS NU"
270 :
```

```
280 POKE SCROLL,24
290 GOSUB 1000:CLS
295 :
340 LOCATE 0,20
345  FOR T=1 TO 200:NEXT T
350 PRINT"DEZE TEKST BLIJFT ECHTER"
360 PRINT"STAAN !!"
370 :
380 POKE SCROLL,25
390 GOSUB 1000
395 :
400 LOCATE 0,0
410 POKE SCROLL,24
420 END
1000 '===========================
1010 '   DEZE SUBROUTINE PRINT
1020 ' STEEDS OP REGEL  24 EEN
1030 '         RIJ "X"
1040 '===========================
1050 FOR G=1 TO 23
1060 LOCATE 0,23
1065  BEEP
1070  PRINT STRING$(39,"X");
1075  FOR T=1 TO 50:NEXT T
1080 NEXT G
1090 RETURN
```

Scherm wissen

In een eerder verschenen deeltje van 'Truuks en Tips' stonden verscheidene routines om het scherm te wissen. Déze routine heeft echter als voordeel dat hij zich aanpast aan de schermmode (0 of 1) en de breedte van het scherm. Hij is dus bijna altijd te gebruiken als een alternatieve manier om het scherm te wissen.

```
100 '===========================
110 '
120 '        SCHERM WISSEN
130 '
140 '===========================
150 '
160 ' SCREEN 0  EN SCREEN 1
170 '
180 '===========================
190 :
200 WI=PEEK(&HF3B0)
205 KEY OFF
210 SOUND7,&B11100111
220 SOUND8,&B00010000
230 SOUND11,100
240 SOUND12,55
250 SOUND13,0
260 POKE &HF3B1,25
270 FOR X=1 TO 12
280 SOUND6,X*2
290  LOCATE0,X-1
300  PRINT STRING$(WI,192);
310  LOCATE0,24-X
320  PRINT STRING$(WI,195);
340  LOCATE0,X-1
350  PRINT STRING$(WI,32);
360  LOCATE0,24-X
370  PRINT STRING$(WI,32);
```

```
380 NEXT X
390 DEFUSR=&H90:X=USR(0)
400 POKE &HF3B1,24
405 CLS
410 :
```

Nogmaals 'Cload en runnen'

In 'Truuks en Tips' deel 3 stond een programma dat een Cload programma automatisch laadde en startte. Helaas staat er een fout in dat programma zodat het soms wel en soms niet werkt. Nog vreemder is dat wel veel kommerciële programma's van bijvoorbeeld Aackosoft dezelfde routine gebruiken met dezelfde fout erin (!?) Redenen genoeg dus om hier de korrekte listing één keer af te drukken:

```
1 REM ************************
2 REM *                      *
3 REM * CLOAD + RUN.....     *
4 REM *                      *
5 REM ************************
6 :
10 KEYOFF:COLOR15,4,4:SCREEN 0:LOCATE 10
,10:PRINT:LOADING...."
20   Q$="RUN"+CHR$(13):FORQ=-1040TO-1000
:POKEQ,0:NEXT:POKE-3078,240:POKE-3077,25
1:POKE-3080,240+LEN(Q$):POKE-3079,251
30   FORQ=1TOLEN(Q$):POKEQ-1041,ASC(MID$
(Q$,Q,1)):NEXT:CLOAD
```

Sprite editor

Vrijwel iedereen die op z'n MSX zelf programmeert heeft wel één of meerdere Sprite-editors. Waarom nu deze? Welnu, deze sprite-editor is speciaal bedoeld voor kassette-gebruikers. Hij is namelijk heel kort en daarom ook zó van kassette ingeladen. De meeste sprite-editors zijn erg lang en lenen zich daarom niet om 'even een sprite te ontwerpen'. De gebruiksaanwijzing is simpel. Na het runnen verschijnt er een 16x 16 matrix op het scherm met een klein vierkantje erin. Dit vierkantje is de kursor. De volgende toetsen kunnen gebruikt worden:

Kursor-toetsen	: beweging van kursor
'F1'	: Punt zetten
'F2'	: Punt weghalen
'F3'	: De bijbehorende data's worden berekend en afgebeeld; ook wordt de sprite op ware grootte afgebeeld.
'Home'-toets	: kursor naar linksboven
'Shift'+'Home'	: wist het matrix

```
10 '-------------------------------'
20 '     KORTE SPRITE EDITOR       '
30 '-------------------------------'
40 ' T R U U K S  &  T I P S   8  '
50 '-------------------------------'
60 '
100 COLOR 1,14,14:CLEAR1000
110 SCREEN 1,2:WIDTH29
120 FORG=1TO8:READA:G$=G$+CHR$(A):NEXT:S
PRITE$(0)=G$
130 DATA 255,129,129,129,129,129,129,255
140 VR=6145+32:KEYOFF
150 A1$="┌─SPRITE───EDITOR─┐":A2$="
|                  |":A3$="└───────
W─────────────┘":PRINTA1$:FORG=1
TO16:PRINTA2$:NEXT:PRINTA3$:'**** iedere
 string is 18 tekens lang ****
160 LOCATE,18:PRINT"'F1'= punt zetten":P
RINT"'F2'= punt weghalen":PRINT"'F3'= da
```

```
ta's":PRINT:PRINT"** (C)1986  MARCEL KRE
EFT **"
170 ONKEYGOSUB330,340,350:KEY(1)ON:KEY(2
)ON:KEY(3)ON
180 XS=24:YS=7
190 PUTSPRITE0,(XS,YS),3
200 Z$=INKEY$:IFZ$=""THEN200
210 IFZ$=CHR$(28)THENXS=XS+8:GOTO280
220 IFZ$=CHR$(29)THENXS=XS-8:GOTO280
230 IFZ$=CHR$(30)THENYS=YS-8:GOTO280
240 IFZ$=CHR$(31)THENYS=YS+8:GOTO280
250 IFZ$=CHR$(11)THEN180
260 IFZ$=CHR$(12)THENRUN
270 GOTO200
280 IF XS>24+15*8THENXS=24
290 IF XS<24      THENXS=24+15*8
300 IF YS>7+15*8 THENYS=7
310 IF YS<7       THENYS=7+15*8
320 GOTO190
330 KEY(1)OFF:KEY(2)OFF:KEY(3)OFF:GOSUB3
60:BEEP:VPOKEVR+X+Y*32,219:Z$=CHR$(28):K
EY(1)ON:KEY(2)ON:KEY(3)ON:RETURN 210
340 KEY(1)OFF:KEY(2)OFF:KEY(3)OFF:GOSUB3
60:BEEP:VPOKEVR+X+Y*32,32  :Z$=CHR$(28):K
EY(1)ON:KEY(2)ON:KEY(3)ON:RETURN 210
350 KEY(1)OFF:KEY(2)OFF:KEY(3)OFF:BEEP:L
OCATE21,0:PRINT"MOMENT":GOSUB370:LOCATE2
1,0:PRINT"      ":KEY(1)ON:KEY(2)ON:KEY(
3)ON:RETURN
360 X=XS/8-1:Y=(YS+1)/8-1:RETURN
370 S$="":FOR Y=0TO7:B$="":FORX=2TO9:IFV
PEEK(VR+X+Y*32)>32THENB$=B$+"1" ELSE B$=
B$+"0"
380 NEXT:B1%(Y)=VAL("&B"+B$):S$=S$+CHR$(
B1%(Y)):NEXT
390 FOR Y=8TO15:B$="":FORX=2TO9:IFVPEEK(
```

```
VR+X+Y*32)>32THENB$=B$+"1" ELSE B$=B$+"0
"
400 NEXT:B2%(Y-8)=VAL("&B"+B$):S$=S$+CHR
$(B2%(Y-8)):NEXT
410 FOR Y=0TO7:B$="":FORX=10TO17:IFVPEEK
(VR+X+Y*32)>32THENB$=B$+"1" ELSE B$=B$+"
0"
420 NEXT:B3%(Y)=VAL("&B"+B$):S$=S$+CHR$(
B3%(Y)):NEXT
430 FOR Y=8TO15:B$="":FORX=10TO17:IFVPEE
K(VR+X+Y*32)>32THENB$=B$+"1" ELSE B$=B$+
"0"
440 NEXT:B4%(Y-8)=VAL("&B"+B$):S$=S$+CHR
$(B4%(Y-8)):NEXT
450 SPRITE$(1)=S$:PUTSPRITE1,(200,148),4
460 X=19:FORY=1TO8 :LOCATEX,Y:PRINTUSING
"###";B1%(Y-1):NEXT
470 X=19:FORY=9TO16:LOCATEX,Y:PRINTUSING
"###";B2%(Y-9):NEXT
480 X=25:FORY=1TO8 :LOCATEX,Y:PRINTUSING
"###";B3%(Y-1):NEXT
490 X=25:FORY=9TO16:LOCATEX,Y:PRINTUSING
"###";B4%(Y-9):NEXT
500 RETURN
```

SCREEN dump op tab-toets

Hoe vaak gebeurt het niet dat u op het beeldscherm belangrijke informatie hebt staan, en dat u genoodzaakt bent om pen en papier te pakken en snel alles op te schrijven terwijl u toch een mooie, dure printer naast uw computer heeft staan?

Vanaf vandaag niet meer getreurd; het volgende programma dat zich, net als bijna alle andere programma's in dit boekje, automatisch op het hoogste punt van het geheugen zet maakt op elk willekeurig moment een dump van SCREEN0 op de printer. Zodra u op de ((tab)) toets drukt wordt het scherm op papier gezet.

Om een toepassing te noemen, zelf gebruik ik het vaak in kombinatie met het assembler/disassembler/debugger pakket CHAMP. Het werkt perfekt samen met dit en vele andere dergelijke utilities.

```
1 '==============================
2 '
3 '     SCREEN DUMP SCREEN 0
5 '
6 '         <TAB> TOETS
8 '
9 '==============================
10 '
15 CLEAR200,(PEEK(&HFC4A)+
      256*PEEK(&HFC4B))-&H40
20 '
27 '
30 SA=PEEK(&HFC4A)+256*PEEK(&HFC4B)
35 SO=SA
40 '
41 DEF FNH(X)=INT(X/256)
42 DEF FNL(X)=X-FNH(X)*256
43 '
46 S1=SA+&HE
50 '
100 '
```

```
110 '
120  READ H$:IF H$="*" THEN 200
130  FOR H=1 TO LEN(H$) STEP 2
140   C$=MID$(H$,H,2)
150   IF C$="  " THEN 180
151   P=VAL("&H"+C$)
152   IF C$="*1" THEN P=FNL(S1)
153   IF C$="*2" THEN P=FNH(S1)
160   POKE SA,P
170   SA=SA+1
180  NEXT H
190 GOTO 120
200 '
210 '======= AKTIVEREN =======
220 '
230 DEF USR=SO:X=USR(0)
240 END
999 '
1000 DATA F3     :'     DI
1010 DATA 3EC3   :'     LD   A,#C3
1020 DATA 32D1FD:'     LD   (#FDD1),A
1030 DATA 21*1*2:'     LD   HL,START
1040 DATA 22D2FD:'     LD   (#FDD2),HL
1050 DATA FB     :'     EI
1060 DATA C9     :'     RET
1070 '           :' START
1080 DATA FE3B   :'     CP   59
1090 DATA C0     :'     RET  NZ
1100 DATA E5     :'     PUSH HL
1110 DATA C5     :'     PUSH BC
1120 DATA 210000:'     LD   HL,0
1130 DATA 011828:'     LD   BC,#2818
1140 DATA CD4A00:' L1  CALL #4A
1150 DATA FE20   :'     CP   #20
1160 DATA 3803   :'     JR   C,L2
1170 DATA CDA500:'     CALL #A5
```

```
1180 DATA 23     :' L2 INC  HL
1190 DATA 10F3   :'    DJNZ L1
1200 DATA 3E0A   :'    LD   A,#0A
1210 DATA CDA500:'     CALL #A5
1220 DATA 3E0D   :'    LD   A,#0D
1230 DATA CDA500:'     CALL #A5
1240 DATA 0628   :'    LD   B,#28
1250 DATA 0D     :'    DEC  C
1260 DATA 20E4   :'    JR   NZ,L1
1270 DATA C1     :'    POP  BC
1280 DATA E1     :'    POP  HL
1290 DATA 33     :'    INC  SP
1300 DATA 33     :'    INC  SP
1310 DATA C9     :'    RET
1320 DATA *
```

Joystick

Bij hoeveel Basic-spelletjes gebeurt het niet dat u moet kiezen; óf joystick óf toetsenbord? Waarom niet én joystick én toetsenbord? Het antwoord is eenvoudig; om meerdere spelregelaars af te vragen, zijn meer Basic-regels nodig en loopt het programma dientengevolge een stuk langzamer. Ter illustratie: de volgende Basic-regel zou nodig zijn om de drie spelregelaars af te vragen:

```
100 J=STICK(0) : IF J=0 THEN J=STICK(1) : IF J=0 THEN J=STICK(2)
```

Bovenstaande regel kan, nadat u eerst het onderstaande programma gerund heeft (of in uw eigen programma ingelast en aangeroepen heeft) vervangen worden door:

```
100 J=USR(0)
```

Joystick 1, joystick 2 en de kursortoetsen worden nu alle drie afgevraagd zonder merkbaar snelheidsverlies.

```
1 '=================================
2 '        JOYSTICK TEST
3 '
4 ' DEZE ROUTINE LEEST ALLE 3
5 ' DE JOYSTICKS TEGELIJK UIT
6 ' EN GEEFT DE WAARDE OVER
7 ' AAN DE 'USR' VARIABELE
8 '
9 '=================================
10 '
15 CLEAR200,(PEEK(&HFC4A)+
         256*PEEK(&HFC4B))-&H40
20 '
27 '
30 SA=PEEK(&HFC4A)+256*PEEK(&HFC4B)
35 SO=SA
100 '
```

```
110 '
120 READ H$:IF H$="*" THEN 200
130  FOR H=1 TO LEN(H$) STEP 2
140   C$=MID$(H$,H,2)
150   IF C$="  " THEN 180
151   P=VAL("&H"+C$)
160   POKE SA,P
170   SA=SA+1
180  NEXT H
190 GOTO 120
200 '
210 '=======    DEMO    =======
220 '
230 DEF USR=SO
240 CLS
250 LOCATE 3,0
255  PRINT"JOYSTICK STATUS: ";
260  PRINT USR(0);" "
265 GOTO 250
270 END
999 '
1000 DATA 3E00  :'      LD A,0
1010 DATA CDD500:'      CALL #D5
1020 DATA FE00  :'      CP 0
1030 DATA 200D  :'      JR NZ,END
1040 DATA 3C    :'      INC A
1050 DATA CDD500:'      CALL #D5
1060 DATA FE00  :'      CP 0
1070 DATA 2005  :'      JR NZ,END
1080 DATA 3E02  :'      LD A,2
1090 DATA CDD500:'      CALL #D5
1100 DATA 2600  :' END: LD H,0
1110 DATA 6F    :'      LD L,A
1120 DATA 22F8F7:'      LD (#F7F8),HL
1130 DATA 3E02  :'      LD A,2
1140 DATA 3263F6:'      LD (#F663),A
```

```
1150 DATA C9     :'         RET
1160 DATA *
```

Fire

Deze routine sluit aan bij de vorige routine en leest de drie vuurknoppen tegelijk uit (joy 1, joy 2 en de spatiebalk). In feite vervangt deze routine de volgende Basic-regel:

100 S=STRIG(0) : IF S=0 THEN S=STRIG(1) : IF S=0 THEN S=STRIG(2)

Alleen de hoofd vuurknop wordt door deze routine uitgelezen, de extra funktietoets op sommige joysticks wordt buiten beschouwing gelaten.

Bovenstaande regel kan door dit programma vervangen worden door:

100 S=USR1(0)

Opmerking: Naar gelang de toestand van de vuurknoppen kan S de waarde 0 of 255 krijgen.

0 = geen vuurknop ingedrukt
255= wèl een vuurknop ingedrukt

```
1 '=================================
2 '        VUURKNOP TEST
3 '
4 ' DEZE ROUTINE LEEST ALLE 3
5 ' DE VUURKNOPPEN TEGELIJK UIT
6 ' EN GEEFT DE WAARDE (0 OF 255)
7 ' OVER AAN DE 'USR' VARIABELE
8 '
9 '=================================
10 '
15 CLEAR200,(PEEK(&HFC4A)+
          256*PEEK(&HFC4B))-&H40
20 '
27 '
30 SA=PEEK(&HFC4A)+256*PEEK(&HFC4B)
35 SO=SA
100 '
```

```
110 '
120 READ H$:IF H$="*" THEN 200
130  FOR H=1 TO LEN(H$) STEP 2
140   C$=MID$(H$,H,2)
150   IF C$="  " THEN 180
151   P=VAL("&H"+C$)
160   POKE SA,P
170   SA=SA+1
180  NEXT H
190 GOTO 120
200 '
210 '=======    TEST    =======
220 '
230 DEF USR=SO
240 CLS
250 LOCATE 3,0
255  PRINT"VUURKNOP STATUS: ";
260  PRINT USR(0);" "
265 GOTO 250
270 END
999 '
1000 DATA 3E00  :'      LD A,0
1010 DATA CDD800:'      CALL #D8
1020 DATA FE00  :'      CP 0
1030 DATA 200D  :'      JR NZ,END
1040 DATA 3C    :'      INC A
1050 DATA CDD800:'      CALL #D8
1060 DATA FE00  :'      CP 0
1070 DATA 2005  :'      JR NZ,END
1080 DATA 3E02  :'      LD A,2
1090 DATA CDD800:'      CALL #D8
1100 DATA 2600  :' END: LD H,0
1110 DATA 6F    :'      LD L,A
1120 DATA 22F8F7:'      LD (#F7F8),HL
1130 DATA 3E02  :'      LD A,2
1140 DATA 3263F6:'      LD (#F663),A
```

```
1150 DATA C9      :'        RET
1160 DATA *
```

Tab-reset

In de 'Truuks en Tips' reeks hebben al verscheidene programmaatjes gestaan om op één of andere manier uit een beveiligd progamma te springen. Het kon echter voorkomen dat het beveiligde programma ook daartegen beveiligd was. In deze versie is die kans een stuk kleiner geworden, daar deze routine op *twee totaal onafhankelijke* interrupts staat.

Het gebruik is simpel: u laadt en runt dit programma. Vervolgens laadt u een ander programma. Mocht u met de kombinatie ((CONTROL)) + ((STOP)) niet uit het programma kunnen springen, aktiveer deze routine dan door op de ((TAB))-toets te drukken. U wordt op de bekende manier met "OK" begroet en kunt weer kommando's invoeren.

In sommige gevallen kunt u ook machinetaalprogramma's die 'vastlopen' onderbreken door op ((TAB)) te drukken.

```
1 '================================
2 '
3 '           SUPER RESET
4 '
5 ' RESET NAAR DE BASIC EDITOR
6 '
7 '           <TAB> TOETS
8 '
9 '================================
10 '
15 CLEAR200,(PEEK(&HFC4A)+
      256*PEEK(&HFC4B))-&H40
20 '
27 '
30 SA=PEEK(&HFC4A)+256*PEEK(&HFC4B)
35 SO=SA
40 '
41 DEF FNH(X)=INT(X/256)
42 DEF FNL(X)=X-FNH(X)*256
43 '
```

```
45 PATCH=SA+&H34
46 S1   =SA+&H22
47 S2   =SA+&H28
50 '
100 '
110 '
120  READ H$:IF H$="*" THEN 200
130  FOR H=1 TO LEN(H$) STEP 2
140   C$=MID$(H$,H,2)
150   IF C$="  " THEN 180
151   P=VAL("&H"+C$)
152   IF C$="*1" THEN P=FNL(PA)
153   IF C$="*2" THEN P=FNH(PA)
154   IF C$="*3" THEN P=FNL(S1)
155   IF C$="*4" THEN P=FNH(S1)
156   IF C$="*5" THEN P=FNL(S2)
157   IF C$="*6" THEN P=FNH(S2)
160   POKE SA,P
170   SA=SA+1
180  NEXT H
190 GOTO 120
200 '
210 '======= AKTIVEREN =======
220 '
230 DEF USR=SO:X=USR(0)
240 END
1000 '
1010 DATA F3     :' DI
1020 DATA 219FFD:' LD HL,HOOK2
1030 DATA 11*1*2:' LD DE,PATCH
1040 DATA 010500:' LD BC,5
1050 DATA EDB0   :' LDIR
1060 '           :'
1070 DATA 3EC3   :' LD A,#C3
1080 DATA 21*3*4:' LD HL,START
1090 DATA 32CCFD:' LD (HOOK1),A
```

```
1100 DATA 22CDFD:' LD (HOOK1+1),HL
1110 DATA 21*5*6:' LD HL,START2
1120 DATA 329FFD:' LD (HOOK2),A
1130 DATA 22A0FD:' LD (HOOK2+1),HL
1140 DATA FB    :' EI
1150 DATA C9    :' RET
1160 :          :' ---START---
1170 DATA FE3B  :' CP #3B
1180 DATA CA9B40:' JP Z,#409B
1190 DATA C9    :' RET
1200 :          :' ---START2---
1210 DATA F5    :' PUSH AF
1220 DATA 2AF8F3:' LD HL,(#F3F8)
1230 DATA 2B    :' DEC HL
1240 DATA 7E    :' LD A,(HL)
1250 DATA FE09  :' CP #09
1260 DATA CA9B40:' JP Z,#409B
1270 DATA F1    :' POP AF
1280 :          :'
1290 DATA 000000:' P A T C H
1300 DATA 0000C9:'
1310 DATA *
```

Toetsrepetitie

De toetsen van het MSX toetsenbord repeteren met een vaste snelheid. Als u een toets ingedrukt houdt, duurt het even alvorens de toets gaat 'lopen'. Bij het normale programmeren is dit heel handig, maar als het om spelletjes gaat die met de kursortoetsen werken, kan dit heel hinderlijk zijn en de uitvoering van het programma vertragen. De volgende routine, die in een ongebruikt stukje RAM staat, laat u zelf de snelheid van de toets-repetitie instellen. Dit gebeurt door middel van POKE &HFC99, s waarbij 's' de snelheid bepaalt (1=heel snel, 4=gemiddeld).

```
100 '================================
105 '
110 '   REPETEER-SNELHEID VAN
115 '
120 '       TOETSEN BEPALEN
125 '
130 ' ==>  POKE &HFC99, SN
136 '
140 '================================
145 '
155 DEF FNH(X)=INT(X/256)
156 DEF FNL(X)=X-FNH(X)*256
160 '
165 SA=2^16+&HFC82
170 SO=SA
175 '
180 '
185 '
190 READ H$:IF H$="*" THEN 235
195  FOR H=1 TO LEN(H$) STEP 2
200   C$=MID$(H$,H,2)
205   IF C$="  " THEN 225
210   P=VAL("&H"+C$)
215   POKE SA,P
220   SA=SA+1
225  NEXT H
```

```
230 GOTO 190
235 '
240 '======= AKTIVEREN =======
245 '
250 SN=3
260  POKE &HFC99,SN
270  POKE &HFDCD,FNL(SO)
280  POKE &HFDCE,FNH(SO)
290  POKE &HFDCC,&HCD
300 END
310 '
1000 DATA F5    :' PUSH  AF
1010 DATA 3A99FC:' LD    A,(#FC99)
1020 DATA 32F7F3:' LD    (#F3F7),A
1030 DATA F1    :'POP   AF
1040 DATA C9    :' RET
1330 '
1340 DATA *
```

Uitbreiding Basic-editor

De Basic-editor van MSX is erg omvangrijk. Maar toch zijn er nog enkele dingen die ontbreken. Eén van die dingen is het samenvoegen van twee regels. In de normale editor dient u de tweede regel opnieuw achter de eerste aan te typen en dan met return in het geheugen te plaatsen. Bovendien moet u de toegevoegde regel nog handmatig te verwijderen. Al met al een omslachtige klus. Met de volgende routine wordt het u wel heel gemakkelijk gemaakt. Met één kommando vervangt u het hele bovenstaande proces. Dit kommando luidt als volgt:

XX=USR9 (regelnummer)

Het regelnummer dat tussen haakjes staat wordt nu gekoppeld aan de voorafgaande regel (met een dubbele punt ertussen), en de regel tussen haakjes wordt verwijderd uit het programma.

```
100 '================================
105 '
110 ' PROGRAMMA REGELS 'MERGEN'
115 '
120 '       dummy= USR9( rn )
125 '
130 '   regel rn is nu gekoppeld
135 '   aan de voorafgaande regel
136 '
140 '================================
145 '
150 CLEAR200,(PEEK(&HFC4A)+
                  256*PEEK(&HFC4B))-&H40
155 '
160 '
165 SA=PEEK(&HFC4A)+256*PEEK(&HFC4B)
170 SO=SA
175 '
180 '
185 '
190 READ H$:IF H$="*" THEN 235
```

```
195  FOR H=1 TO LEN(H$) STEP 2
200   C$=MID$(H$,H,2)
205   IF C$="  " THEN 225
210   P=VAL("&H"+C$)
215   POKE SA,P
220   SA=SA+1
225  NEXT H
230 GOTO 190
235 '
240 '======= AKTIVEREN =======
245 '
250 DEF USR9=SO
255 KEY 1,"X=USR9( )"+CHR$(29)+CHR$(29)
                     +CHR$(18)
260 END
1000 DATA ED5BF8F7:' LD DE,(#F7F8)
1010 DATA CD9542  :' CALL #4295
1020 DATA C0      :' RET NZ
1030 DATA C5      :' PUSH BC
1040 DATA C5      :' PUSH BC
1045 DATA CDEA54  :' CALL #54EA
1050 DATA D1      :' POP DE
1060 DATA 1B      :' DEC DE
1070 DATA 3E3A    :' LD A,":"
1080 DATA 12      :' LD (DE),A
1090 DATA 13      :' INC DE
1100 DATA 2AC2F6  :' LD HL,(VARTAB)
1110 DATA 2B      :' DEC HL
1120 DATA 2B      :' DEC HL
1130 DATA 2B      :' DEC HL
1140 DATA 2290FC  :' LD (#FC90),HL
1150 DATA 37      :' SCF
1160 DATA 3F      :' CCF
1170 DATA ED52    :' SBC HL,DE
1180 DATA 44      :' LD B,H
1190 DATA 4D      :' LD C,L
```

```
1200 DATA E1      :' POP HL
1210 DATA 23      :' INC HL
1220 DATA 23      :' INC HL
1230 DATA 23      :' INC HL
1240 DATA 23      :' INC HL
1250 DATA EDB0    :' LDIR
1260 DATA 2A90FC  :' LD HL,(#FC90)
1270 DATA 2B      :' DEC HL
1280 DATA 22C2F6  :' LD (VARTAB),HL
1290 DATA 22C4F6  :' LD (ARYTAB),HL
1300 DATA 22C6F6  :' LD (STREND),HL
1310 DATA CD5342  :' CALL #4253
1320 DATA C9      :' RET
1330 '
1340 DATA *
```

Smooth scrolling tekst

Hoe vaak hebt u niet met jaloerse ogen naar professionele (machinetaal) software gekeken waaronder het beeld de gebruiksaanwijzing heel vloeiend voorbij scrollt en waarbij u dacht dat dat voor u, een eenvoudige Basic programmeur, onbereikbaar was? Welnu, uw tijd is gekomen. Met onderstaande routine kunt u elke willekeurige tekst van elke willekeurige lengte geheel vloeiend over het scherm laten scrollen. De routine werkt alleen op SCREEN1.

Deze routine is één van de meest komplexe in dit boekje en verdient daarom nadere uitleg om het goed te kunnen benutten.

Tot en met regel 260 leest de machinetaal in en plaatst deze zo hoog mogelijk in het geheugen.
Regel 280 definieert de te scrollen tekst en de regels 290-310 plaatst deze op de juiste plaats in het geheugen. Regel 320 zet achter de tekst het 'eind-van-tekst' -teken. In regel 340-380 wordt de routine door middel van USR(0) geïnitialiseerd. Deze USR-call wist de karakters die hij later gebruikt en zorgt er voor dat de tekst straks bij het begin gaat scrollen.
Regel 410-440 plaatsen de benodigde karakters (dit zijn de karakters met nummer 200 tot en met 232) onder op het scherm. (U kunt ze natuurlijk op een andere plaats zetten indien u dat wenst).
Regel 450-451 zet een tekst ergens anders op het scherm.

Regel 460 is heel belangrijk. Doordat de routine twee verschillende uitstappunten heeft, loopt de timing van het scrollen niet altijd goed en kan het 'stotterend' gaan of er kan een trilling over het scherm lopen. Om dit tegen te gaan dient de poke in regel 460. De juiste waarde is echter afhankelijk van de situatie waarin u de routine gebruikt en kan alleen maar gevonden worden door te proberen. De waarde zal waarschijnlijk altijd liggen tussen 100 en 200. Regel 470 scrollt de tekst steeds één pixel naar links en herhaald dit steeds. Als u het programma runt zult u zien dat de tekst, hoewel het SCREEN1 is, toch op de normale SCREEN0 letterafstand voorbij scrollt. Wilt u dit veranderen dan dient u in de regels 1000 en 1130 de data "06" te veranderen in de gewenste letterbreedte. U dient dan echter ook de TIMING POKE aan te passen.

```
1 '================================
2 '
3 '   smooth-scrolling tekst
4 '
5 '================================
6 '
7 '    (C)1987  MARCEL KREEFT
8 '
9 '================================
10 '
11 COLOR 1,15,15
12 :
15 CLEAR200,(PEEK(&HFC4A)+
      256*PEEK(&HFC4B))-&H2AF
20 :
27 :
30 SA=PEEK(&HFC4A)+256*PEEK(&HFC4B)
35 SO=SA
40 :
41 DEF FNH(X)=INT(X/256)
42 DEF FNL(X)=X-FNH(X)*256
43 :
45 X(1)=SO+&HA3   :' TELLER
46 X(2)=SO+&HA5   :' POINTER
47 X(3)=SO+&HA7   :' BUFFER
48 X(4)=SO+&H1AF  :' TABEL
49 X(5)=SO+&H95   :' CLEAR
50 X(6)=SO+&H75   :' COPY
51 X(7)=SO+&H3D   :' PIXEL
52 X(8)=SO+&H1A7  :' BUFFER+32*8
100 :
110 :
120  READ H$:IF H$="*" THEN 270
130  FOR H=1 TO LEN(H$) STEP 2
140   C$=MID$(H$,H,2)
150   IF C$="  " THEN 250
```

```
160    P=VAL("&H"+C$)
170    IF LEFT$(C$,1)="#" THEN NO=
         VAL(RIGHT$(C$,1)):POKE SA,
         FNL(X(NO)):POKE SA+1,FNH(X(NO))
         :SA=SA+2:GOTO 250
230    POKE SA,P
240    SA=SA+1
250   NEXT H
260  GOTO 120
270  :
272  :
280  TX$="DIT IS EEN DEMONSTRATIE VAN SMO
OTH-SCROLLING-TEXT UIT `TRUUKS & TIPS DE
EL 8' ......."
285  :
286  '===== TEKST IN GEHEUGEN POKEN =====
287  :
290  FOR G=1 TO LEN(TX$)
300   POKE X(4)+G-1,ASC(MID$(TX$,G,1))
310  NEXT G
312  :
315  '======= SLUITGEGEVEN  (255) =======
316  :
320   POKE X(4)+G-1,255
330  :
335  '=============== DEMO ===============
336  :
340  SCREEN 1
350  DEF USR =SO       :' INITIALISATIE
360  DEF USR1=SO+&H13 :' SCROLL 1 PIXEL
370  :
380  X=USR(0): ' INITIALISEREN
385  :
390  :
400  WIDTH 32:KEYOFF
405  :
```

```
410 LOCATE0,20          :' PRINT KAR/SET
420  FOR G=200 TO 231   :' OP SCHERM.
430   PRINTCHR$(G);     :' (200-231)
440  NEXT G
445 :
450 LOCATE 5,10
451 PRINT "(C)1987 MARCEL KREEFT"
452 :
455 '=================================
456 ' TIMING-WAARDE (TEGEN TRILLING)
457 '=================================
458 :
460 POKE &HFF77,170:POKE&HFF78,0
465 :
470 X=USR1(0):GOTO470
998 :
999 :
1000 DATA 3E06  :'    LD    A,6
1010 DATA 32#1  :'    LD    (TELLER),A
1020 DATA CD#5  :'    CALL  CLEAR
1030 DATA 21#4  :'RT  LD    HL,TABEL
1040 DATA 7E    :'    LD    A,(HL)
1050 DATA CD#6  :'    CALL  COPY
1060 DATA 22#2  :'    LD    (POINTER),HL
1070 DATA C9    :'    RET
1080 DATA CD#7  :'    CALL  PIXEL
1090 DATA 3A#1  :'    LD    A,(TELLER)
1100 DATA 3D    :'    DEC   A
1110 DATA 32#1  :'    LD    (TELLER),A
1120 DATA 2015  :'    JR    NZ,W   ;WACHT
1130 DATA 3E06  :'    LD    A,6
1140 DATA 32#1  :'    LD    (TELLER),A
1150 DATA 2A#2  :'    LD    HL,(POINTER)
1160 DATA 23    :'    INC   HL
1170 DATA 7E    :'    LD    A,(HL)
1180 DATA FEFF  :'    CP    #FF
```

```
1190 DATA 28DB  :'    JR   Z,RT   ;RESET
1200 DATA CD#6  :'    CALL COPY
1210 DATA 22#2  :'    LD   (POINTER),HL
1220 DATA C9    :'    RET
1230 DATA 2A77FF:'W   LD   HL,(#FF77)
1240 DATA 2B    :'W1  DEC  HL
1250 DATA 7C    :'    LD   A,H
1260 DATA B5    :'    OR   L
1270 DATA 20FB  :'    JR   NZ,W1
1280 DATA C9    :'    RET
1290 DATA 214006:'#7  LD   HL,1600
1300 DATA 010801:'    LD   BC,256+8
1310 DATA 11#3  :'    LD   DE,BUFFER
1320 DATA CD5900:'    CALL #59
1330 DATA 21#8  :'    LD   HL,BUF+32*8
1340 DATA 0608  :'    LD   B,8
1350 DATA 37    :'    SCF
1360 DATA 3F    :'    CCF
1370 DATA C5    :'L1  PUSH BC
1380 DATA E5    :'    PUSH HL
1390 DATA 0620  :'    LD   B,32
1400 DATA 7E    :'L2  LD   A,(HL)
1410 DATA 17    :'    RLA
1420 DATA 77    :'    LD   (HL),A
1430 DATA 2B    :'    DEC  HL
1440 DATA 2B    :'    DEC  HL
1450 DATA 2B    :'    DEC  HL
1460 DATA 2B    :'    DEC  HL
1470 DATA 2B    :'    DEC  HL
1480 DATA 2B    :'    DEC  HL
1490 DATA 2B    :'    DEC  HL
1500 DATA 2B    :'    DEC  HL
1510 DATA 10F3  :'    DJNZ L2
1520 DATA 37    :'    SCF
1530 DATA 3F    :'    CCF
1540 DATA E1    :'    POP  HL
```

```
1550 DATA 23    :'    INC  HL
1560 DATA C1    :'    POP  BC
1570 DATA 10E8  :'    DJNZ L1
1580 DATA 21#3  :'    LD   HL,BUFFER
1590 DATA 114006:'    LD   DE,1600
1600 DATA 010801:'    LD   BC,256+8
1610 DATA CD5C00:'    CALL #5C
1620 DATA C9    :'    RET
1630 DATA E5    :'#6  PUSH HL
1640 DATA C5    :'    PUSH BC
1650 DATA D5    :'    PUSH DE
1660 DATA 37    :'    SCF
1670 DATA 3F    :'    CCF
1680 DATA 6F    :'    LD   L,A
1690 DATA 2600  :'    LD   H,0
1700 DATA 29    :'    ADD  HL,HL
1710 DATA 29    :'    ADD  HL,HL
1720 DATA 29    :'    ADD  HL,HL
1730 DATA 114007:'    LD   DE,ADRES
1740 DATA 0608  :'    LD   B,8
1750 DATA CD4A00:'S   CALL RVRAM
1760 DATA EB    :'    EX   DE,HL
1770 DATA CD4D00:'    CALL WVRAM
1780 DATA EB    :'    EX   DE,HL
1790 DATA 13    :'    INC  DE
1800 DATA 23    :'    INC  HL
1810 DATA 10F4  :'    DJNZ S
1820 DATA D1    :'    POP  DE
1830 DATA C1    :'    POP  BC
1840 DATA E1    :'    POP  HL
1850 DATA C9    :'    RET
1860 DATA 214006:'#5  LD   HL,1600
1870 DATA 06FF  :'    LD   B,#FF
1880 DATA 3E00  :'    LD   A,0
1890 DATA CD4D00:'SS  CALL WVRAM
1900 DATA 23    :'    INC  HL
```

```
1910 DATA 10FA   :'   DJNZ SS
1920 DATA C9     :'   RET
1930 DATA *
```

80 kolommen utility

Dit programma stelt u in staat op het MSX scherm tussen de 32 en 128 tekens per regel te krijgen, zonder hardwarematige ingrepen aan uw computer. Enige vereiste is dat u een 64kB MSX1 computer heeft.

Dit programma is het langste programma in dit boekje maar zal voor velen ook het meest interessant zijn. In ieder geval zal het het intypen waard zijn. U kunt namelijk kiezen uit 32, 37, 42, 51, 64, 85 of 128 tekens per regel!! Het programma werkt op het grafische scherm (SCREEN2) van uw MSX-computer en kan daardoor met de grafische kommando's (LINE, CIRCLE enzovoort) samenwerken.

Regel 690 tot en met 790 bevat een subroutine die aangeroepen moet worden als u tekst op het grafische scherm wilt printen. Bij deze aanroep moet u van te voren een aantal variabelen gedefinieerd hebben. Deze zijn:

X% =X-positie op het scherm (0-255)
Y% =Y-positie op het scherm (0-190)
C% =Kleur van de te printen tekst
T% =De te printen tekst
A% =De afstand tussen het begin van de eerste en het begin van de daaropvolgende letter

A%=2 ; 128 tekens per regel
A%=3 ; 85 tekens per regel
A%=4 ; 64 tekens per regel
A%=5 ; 51 tekens per regel
A%=6 ; 42 tekens per regel
A%=7 ; 37 tekens per regel
A%=8 ; 32 tekens per regel

Als A% kleiner dan 5 is, wordt een smalle karakterset ingeschakeld. Is A% 5 of groter dan wordt de standaard karakterset gebruikt. Hoe kleiner A% wordt, hoe minder goed leesbaar de tekst wordt. Tot A%=3 is de tekst nog redelijk goed leesbaar. Als men echter met 128 tekens per regel werkt, is de tekst nog maar nauwelijks leesbaar. Het enige nuttige doel dat ik met voor deze mode kan indenken is bijvoorbeeld het tonen van een lay-out van een brief. Er kan dan nog een menu of iets dergelijks naast de brief staan.

Vanaf regel 1000 kunt u uw eigen programma schrijven. In de listing heb ik hier een demonstratie-programmaatje aangeplakt.

Om de werking van het programma te verduidelijken volgt hier een voorbeeld: stel we willen op positie 75, 150 de tekst "MSX Truuks en Tips deel 8" in 85-koloms-mode op het scherm hebben. De volgende regel zou dan nodig zijn:

1010 X%=75: Y%=150: T$="MSX Truuks en Tips deel 8" :A%=3: C%=1: GOSUB 700

Enkele opmerkingen:

- Alvorens u de subroutine op regel 700 voor de eerste keer aanroept, moet u eerst een file naar het scherm geopend hebben. Dit doet u door de volgende regel:

  1000 SCREEN 2:OPEN"GRP:"AS 1

- De subroutine verandert geen enkele variabele. Ook X%, Y%, C%, A% en T$ blijven ongewijzigd.
- Hoe dichter de letter-afstand, hoe sneller (verhoudingsgewijs) de tekst geprint wordt. Voor A%=2, 3, 4 geldt dat de tekst sneller geprint wordt dan normaal de tekst op SCREEN2 geprint wordt.
- Deze routine werkt alléén op MSX1 computers met 64K RAM.

```
0    '-----------------------------------
1  '   -- 80 kolommen MSX V1.2 --
2  '
3  '  (C) 1986 === Marcel Kreeft
4  '-----------------------------------
5 :
6 CLS:PRINT"-moment-"
7 :
8 POKE&H1B0,255:IFPEEK(&H1B0)=255 THEN 1
000
9 :
10 DATA f3,db,a8,32,2,d2,cb,3f,cb,3f,cb,
3f,cb,3f,32,1,d2,cb,3f,cb,3f,32,0,d2,21,
0,0,3e,0
20 DATA cd,c,0,f3,5f,3a,0,d2,cd,14,0,f3,
```

```
23,7c,fe,80,20,ec,3a,1,d2,47,3a,2,d2,80,
d3,a8,fb,c9
90 :
91 '
92 '     MACHINETAAL AKTIVEREN
93 '
94 :
100 FORG=&HC000TO&HC03A:READH$:POKEG,VAL
("&H"+H$):NEXT:DEFUSR=&HC000:X=USR(0)
101 :
102 POKE &H447,0 :' NOODZAKELIJKE
103 POKE &H448,0 :' WIJZIGING IN DE
104 POKE &H449,0 :' CTRL+STOP ROUTINE
105 :
110 '
115 '     KARAKTER-DATA INLEZEN
120 '
125 :
130 RESTORE 215
135 FOR G=1 TO &H1D0
140  READ H$
145  POKE &H1CC6+G,VAL("&H"+H$)
150 NEXT G
155 '
160 FOR G=0 TO &HCF
165  POKE &H1EC7+G,PEEK(&H1DC7+G)
170 NEXT G
175 '
180 RESTORE 520
185 FOR G=1 TO &HF8
190  READ H$
195  POKE &H1DC6+G,VAL("&H"+H$)
200 NEXT G
201 GOTO 1000
205 '
210 '
```

```
215 DATA 40,40,40,40,0,40,0,0        :' !
220 DATA A0,A0,0,0,0,0,0,0           :' "
225 DATA 40,E0,40,E0,40,0,0,0        :' #
230 DATA 40,E0,C0,60,E0,40,0,0       :' $
235 DATA A0,20,40,40,80,A0,0,0       :' %
240 DATA 40,E0,E0,40,E0,E0,60,0      :' &
245 DATA 20,40,0,0,0,0,0,0           :' '
250 DATA 20,40,40,40,40,20,0,0       :' (
255 DATA 40,20,20,20,20,40,0,0       :' )
260 DATA 0,A0,40,E0,40,A0,0,0        :' *
265 DATA 0,0,40,E0,40,0,0,0          :' +
270 DATA 0,0,0,0,20,40,0,0           :' ,
275 DATA 0,0,60,0,0,0,0,0            :' -
280 DATA 0,0,0,0,0,40,0,0            :' .
285 DATA 20,20,40,40,80,80,0,0       :' /
290 DATA 40,E0,A0,A0,E0,40,0,0       :' 0
295 DATA 40,40,40,40,40,40,0,0       :' 1
300 DATA 60,20,60,40,40,60,0,0       :' 2
305 DATA 60,20,60,20,20,60,0,0       :' 3
310 DATA 80,A0,A0,E0,20,20,0,0       :' 4
315 DATA 60,40,60,20,20,60,0,0       :' 5
320 DATA 20,40,40,60,60,60,0,0       :' 6
325 DATA 60,20,60,20,20,20,0,0       :' 7
330 DATA 40,A0,40,A0,A0,40,0,0       :' 8
335 DATA 60,60,60,20,20,60,0,0       :' 9
340 DATA 0,20,0,0,20,0,0,0           :' :
345 DATA 0,20,0,0,20,40,0,0          :' ;
350 DATA 20,40,80,40,20,0,0,0        :' <
355 DATA 0,E0,0,0,E0,0,0,0           :' =
360 DATA 80,40,20,40,80,0,0,0        :' >
365 DATA 60,20,60,40,0,40,0,0        :' ?
370 DATA 40,A0,E0,20,A0,E0,0,0       :' @
375 DATA 0,40,20,20,60,60,0,0        :' a
380 DATA 80,80,80,E0,A0,E0,0,0       :' b
385 DATA 0,60,40,40,40,60,0,0        :' c
390 DATA 20,20,20,E0,A0,E0,0,0       :' d
```

```
395 DATA 0,20,60,60,40,20,0,0    :' e
400 DATA 0,60,40,60,40,40,0,0    :' f
405 DATA 0,E0,A0,E0,20,20,60,0   :' g
410 DATA 80,80,E0,A0,A0,A0,0,0   :' h
415 DATA 40,0,40,40,40,40,0,0    :' i
420 DATA 40,0,40,40,40,40,C0,0   :' j
425 DATA 80,80,A0,C0,A0,A0,0,0   :' k
430 DATA 40,40,40,40,40,60,0,0   :' l
435 DATA 0,A0,E0,E0,A0,A0,0,0    :' m
440 DATA 0,E0,A0,A0,A0,A0,0,0    :' n
445 DATA 0,E0,A0,A0,A0,E0,0,0    :' o
450 DATA 0,E0,A0,E0,80,80,0,0    :' p
455 DATA 0,60,60,60,20,20,0,0    :' q
460 DATA 0,60,40,40,40,40,0,0    :' r
465 DATA 0,60,40,60,20,60,0,0    :' s
470 DATA 40,60,40,40,40,20,0,0   :' t
475 DATA 0,A0,A0,A0,A0,40,0,0    :' u
480 DATA 0,A0,A0,E0,E0,40,0,0    :' v
485 DATA 0,A0,A0,E0,E0,A0,0,0    :' w
490 DATA 0,A0,E0,40,A0,A0,0,0    :' x
495 DATA 0,A0,A0,E0,20,E0,0,0    :' y
500 DATA 0,60,20,40,40,60,0,0    :' z
505 '
515 '
520 DATA 40,A0,A0,E0,A0,A0,0,0   :' A
525 DATA C0,A0,C0,A0,A0,C0,0,0   :' B
530 DATA 40,A0,80,80,A0,40,0,0   :' C
535 DATA C0,A0,A0,A0,A0,C0,0,0   :' D
540 DATA E0,80,C0,80,80,E0,0,0   :' E
545 DATA E0,80,E0,80,80,80,0,0   :' F
550 DATA 40,A0,80,E0,A0,40,0,0   :' G
555 DATA A0,A0,E0,A0,A0,A0,0,0   :' H
560 DATA 40,40,40,40,40,40,0,0   :' I
565 DATA 20,20,20,A0,A0,40,0,0   :' J
570 DATA A0,A0,C0,C0,A0,A0,0,0   :' K
575 DATA 80,80,80,80,80,E0,0,0   :' L
```

```
580 DATA A0,E0,E0,E0,A0,A0,0,0   :' M
585 DATA E0,A0,A0,A0,A0,A0,0,0   :' N
590 DATA 40,A0,A0,A0,A0,40,0,0   :' O
595 DATA C0,A0,A0,C0,80,80,0,0   :' P
600 DATA E0,A0,A0,A0,E0,E0,20,0  :' Q
605 DATA E0,A0,A0,C0,C0,A0,0,0   :' R
610 DATA 60,80,40,20,20,C0,0,0   :' S
615 DATA E0,40,40,40,40,40,0,0   :' T
620 DATA A0,A0,A0,A0,A0,E0,0,0   :' U
625 DATA A0,A0,A0,A0,A0,40,0,0   :' V
630 DATA A0,E0,E0,E0,E0,40,0,0   :' W
635 DATA A0,A0,40,40,A0,A0,0,0   :' X
640 DATA A0,A0,A0,40,40,40,0,0   :' Y
645 DATA E0,20,40,40,80,E0,0,0   :' Z
650 DATA E0,80,80,80,80,E0,0,0   :' [
655 DATA 0,80,C0,40,60,20,0,0    :' \
660 DATA E0,20,20,20,20,E0,0,0   :' ]
665 DATA 40,E0,40,40,40,40,0,0   :' ^
670 DATA 0,0,0,0,0,0,E0,0        :' _
680 '
685 '
690 '============================
695 ' SUBROUTINE 80-KOLOMMEN
700 '============================
705 '
710 '         GOSUB 700
715 '
720 '   X% =  XPOS     Y% = YPOS
725 '   T$ = "TEKST"   C% = KLEUR
730 '   A% = AFSTAND (2,3,4,5,6,7,8)
735 '============================
740 '
745 POKE&HF91F,INT(INP(&HA8)/64)
750 POKE&H1542,3:POKE&H1540,7
755 POKE&H1575,A%:IFA%>4THENPOKE&HF91F,0
:POKE&H1542,8:POKE&H1540,8
```

```
760  POKE &HFCB7,X%      :' GRPACX
765  POKE &HFCB9,Y%      :' GRPACY
770  POKE &HF3E9,C%      :' FORCLR
775  PRINT#1,T$
780 POKE&H1542,8:POKE&H1540,8
785 POKE&HF91F,0:POKE&H1575,6
790 RETURN
795 :
800 :
990 '
991 '=================================
992 '  vanaf regel 1000 uw eigen
993 '  programma's. voordat U de eerste
994 '  keer 'GOSUB 700' gebruikt eerst:
995 '  'SCREEN 2:OPEN"GRP:"AS1'
996 '=================================
997 '
1000 '
1001 '
1002 '  DEMONSTRATIE PROGRAMMA 80KAR
1003 '
1004 '
1005 '
1010 COLOR1,15:SCREEN2:OPEN"grp:"AS1
1020 C%=12:X%=20:Y%=1:A%=3:T$="*** 80 KO
LOMMEN TOOLKIT voor MSX 64 Kram. Ontworp
en door Marcel Kreeft ***":GOSUB 700
1030 C%=1:X%=10
1040 Y%=20:FOR A%=2TO18:LINE(0,Y%-2)-(25
5,Y%-2),6:Y%=Y%+9:NEXT
1050 Y%=20:FOR A%=2TO17:T$="Dit is een D
EMO":GOSUB 700:Y%=Y%+9:NEXT
1060 Y%=20:X%=155:C%=4:A%=5:T$="128 kar
per regel":GOSUB 700
1070 Y%=29:T$=" 85 kar per regel":GOSUB
700
```

```
1080 Y%=38:T$=" 64 kar per regel":GOSUB
700
1090 Y%=47:T$=" 51 kar per regel":GOSUB
700
1100 Y%=56:T$=" 42 kar per regel":GOSUB
700
1110 Y%=65:T$=" 37 kar per regel":GOSUB
700
1120 Y%=74:T$=" 32 kar per regel":GOSUB
700
1130 Y%=80:T$="  .... ":GOSUB 700
1140 GOSUB 2000
1150 SCREEN 0:CLOSE:END
1160 '
1170 ' WACHT OP TOETSDRUK
1180 '
2000 X%=95:Y%=185:A%=4:C%=12:T$="DRUK EE
N TOETS":GOSUB 700
2010 IFINKEY$=""THEN2010
2020 RETURN
```

ROM wordt RAM

Het verschil tussen ROM en RAM geheugen is dat u het ROM met de beste wil van de wereld niet kunt veranderen, en het RAM wél. Logischerwijs worden uw programma's in het RAM opgeslagen en staat de software waar u de computer mee opstart in het ROM (denk hierbij aan het ROM-programma dat u in Basic laat programmeren).
Als we die Basic-ROM zouden kunnen veranderen, zouden we daar heel wat leuke dingen mee kunnen doen.
Welnu, onderstaand programma (dat overigens alleen op MSX1 computers met minstens 64 K RAM werkt) maakt het mogelijk om de Basic/Bios ROM zonder blijvende gevolgen uiteraard, toch te kunnen veranderen. De techniek is als volgt:

een 64K MSX1 computer heeft 4 slots (0-3). In één ervan (slot 0) zit 32K Basic ROM. In een ander slot zit 64K RAM. Omdat de Z-80, de CPU van MSX, maar ten hoogste 64K tegelijk kan besturen, moet deze een selektie maken. Standaard wordt 32K ROM geselekteerd (zodat u in Basic kunt programmeren) en 32K RAM (om uw Basic-programma's in op te slaan).

Dit programma maakt eerst een kopie van het ROM in het overgebleven 32K RAM en selekteert vervolgens 64K RAM. Omdat de Basic-ROM nu in het RAM staat kunt u deze met POKE gewoon veranderen.

Verscheidene programma's maken gebruik van deze techniek.
De routine is als volgt:

```
1 '==============================
2 '   ROM WORDT RAM  (64K MSX1)
3 '
4 '   COPIEERT HET ROM GEHEUGEN
5 '   IN HET  ONDERLIGGENDE RAM
6 '   EN SCHAKELT  HET RAM  IN.
7 '
8 '!! SLECHTS 1 MAAL RUNNEN  !!
9 '==============================
10 '
27 '
30 SA=&HD000
```

```
35 SO=SA
100 '
110 '
120 READ H$:IF H$="*" THEN 200
130  FOR H=1 TO LEN(H$) STEP 2
140   C$=MID$(H$,H,2)
150    IF C$=" " THEN 180
151    P=VAL("&H"+C$)
160    POKE SA,P
170    SA=SA+1
180  NEXT H
190 GOTO 120
200 '
210 '======= AKTIVEREN =======
220 '
230 DEF USR=SO
240 CLS
250 LOCATE 13,0
255  PRINT"MOMENTJE..."
260  X=USR(0)
265 LOCATE 2,2
270 PRINT "HET ROM IS NU RAM GEWORDEN"
271 '
272 POKE &H447,0 :' NOODZAKELIJKE
273 POKE &H448,0 :' WIJZIGING IN DE
274 POKE &H449,0 :' CTRL+STOP ROUTINE
275 '
280 END
999 '
1000 DATA F3    :'    DI
1010 DATA DBA8  :'    IN   A,(#A8)
1020 DATA 3202D1:'    LD   (#D102),A
1030 DATA CB3F  :'    SRL  A
1040 DATA CB3F  :'    SRL  A
1050 DATA CB3F  :'    SRL  A
1060 DATA CB3F  :'    SRL  A
```

```
1070 DATA 3201D1:'      LD   (#D101),A
1080 DATA CB3F   :'      SRL  A
1090 DATA CB3F   :'      SRL  A
1100 DATA 3200D1:'      LD   (#D100),A
1110 DATA 210000:'      LD   HL,0
1120 DATA 3E00   :'L1   LD   A,0
1130 DATA CD0C00:'      CALL #0C
1140 DATA F3     :'      DI
1150 DATA 5F     :'      LD   E,A
1160 DATA 3A00D1:'      LD   A,(#D100)
1170 DATA CD1400:'      CALL #14
1180 DATA F3     :'      DI
1190 DATA 23     :'      INC  HL
1200 DATA 7C     :'      LD   A,H
1210 DATA FE80   :'      CP   #80
1220 DATA 20EC   :'      JR   NZ,L1
1230 DATA 3A01D1:'      LD   A,(#D101)
1240 DATA 47     :'      LD   B,A
1250 DATA 3A02D1:'      LD   A,(#D102)
1260 DATA 80     :'      ADD  A,B
1270 DATA D3A8   :'      OUT  (#A8),A
1280 DATA FB     :'      EI
1290 DATA C9     :'      RET
1300 DATA *
```

```
300 DATA NEXT zonder FOR
310 DATA fout in syntax
320 DATA RETURN zonder GOSUB
330 DATA data's op!
340 DATA foute functie aanroep
350 DATA te hoog getal
360 DATA geheugen vol
370 DATA regel bestaat niet
380 DATA array-nummer fout
390 DATA array bestaat al
400 DATA deling door 0
410 DATA direkt fout
420 DATA type foutje
430 DATA stringruimte vol
440 DATA string te lang
450 DATA string te complex
460 DATA kan niet verdergaan
470 DATA functie bestaat niet
480 DATA randap. I/O fout
490 DATA VERIFY fout
500 DATA geen RESUME
510 DATA RESUME zonder
520 DATA onprintbaar
530 DATA operand ontbreekt
540 DATA regel te lang
550 DATA veld te lang
560 DATA interne fout!!
570 DATA fout file nummer
580 DATA file niet gevonden
590 DATA file is al open
600 DATA ingave te lang
610 DATA foute filenaam
620 DATA direkt commando in file
630 DATA alleen sequentionele I/O
640 DATA file is dicht
650 DATA ***
```

```
660 :
670 :
680 CLS
690 PRINT"------------------------------
-------"
700 PRINT"      NEDERLANDSE FOUTMELDINGE
N"
710 PRINT:PRINT"------------------------
-------------"
720 PRINT"28815 bytes free":PRINT:NEW
730 :
740 :
750 RESTORE 870
760 FOR G=&HE000 TO &HE03A
770  READ A$
780  POKE G,VAL("&H"+A$)
790 NEXT
800 :
810 DEF USR=&HE000
820 DUMMY=USR(0)
830 :
840 RETURN
850 :
860 :
870 DATA F3,DB,A8,32,2,E2,CB,3F,CB,3F,CB
,3F,CB,3F,32,1,E2,CB,3F,CB,3F,32,0,E2,21
,0,0,3E,0
880 DATA CD,C,0,F3,5F,3A,0,E2,CD,14,0,F3
,23,7C,FE,80,20,EC,3A,1,E2,47,3A,2,E2,80
,D3,A8,FB,C9
```

Nieuwe funkties

Sprite-funkties

Wat ik naar mijn mening te weinig tegen kom in Basic-programma's is het gebruik van het DEF FN statement. Dat is jammer want met dit statement kan men in feite de Basic kommando set uitbreiden!

Dit programma illustreert het FN kommando door middel van een viertal SPRITE funkties:

FN SXPR(n)	; geeft de X-koördinaat van sprite n
FN YSPR(n)	; geeft de Y-koördinaat van sprite n
FN KSPR(n)	; geeft de kleur van sprite n
FN PSPR(n)	; geeft het pattern-nummer van sprite n

```
100 '=========================
110 '
120 '   NIEUWE FUNCTIES VOOR
130 '
140 '      S P R I T E S
150 '
160 '=========================
170 '
180 '** X-COORDINAAT **
190 '
200 DEFFN XSPR(SN)=VPEEK(&H1B01+4*SN)
210 '
220 '** Y-COORDINAAT **
230 '
240 DEFFN YSPR(SN)=VPEEK(&H1B00+4*SN)
250 '
260 '** KLEUR SPRITE **
270 '
280 DEFFN KSPR(SN)=VPEEK(&H1B03+4*SN)
290 '
300 '** PATTERN NUM. **
310 '
320 DEFFN PSPR(SN)=VPEEK(&H1B02+4*SN)
```

```
330 '
340 '** DEMONSTRATIE **
345 '
350 SCREEN 1,1:C=1:X=10:Y=50
360 SPRITE$(5)=STRING$(8,255)
370 PUT SPRITE 5,(X,Y),C
380 LOCATE 3,0:PRINT"KLEUR=";
390                PRINT FN KSPR(5)" "
400 '
410 LOCATE 3,1:PRINT"XPOS =";
420                PRINT FN XSPR(5)" "
430 '
440 LOCATE 3,2:PRINT"YPOS =";
450                PRINT FN YSPR(5)" "
460 '
470 X=X+2:IF X>255 THEN X=1
480 Y=Y+4:IF Y>190 THEN Y=1
490 C=C+1:IF C=15  THEN C=1
500 '
530 GOTO 370
```

Stringfunkties

Dit programma geeft vijf nieuwe STRING-funkties.

FN R$(X$,a)	; geeft een string als X$ met a tekens RECHTS afgekapt.
FN L$(X$,a)	; geeft een string als X$ met a tekens LINKS afgekapt.
FN M$(X$,a,b)	; geeft een string als X$ maar verwijdert vanaf positie a, b tekens uit X$.
FN C$(X$,a)	; centreert X$ op regellengte a door links met spaties aan te vullen.
FN SC$(a,b)	; geeft een string van lengte b met getal a rechts opgevuld met nullen. (Bijvoorbeeld FN SC$(25,4) geeft "0025")

```
100 '===========================
110 '
120 '  NIEUWE FUNCTIES VOOR
130 '
140 '     S T R I N G S
150 '
160 '===========================
170 '
180 '** RECHTS AFKAPPEN **
190 '
200 DEFFN R$(X$,A)=LEFT$(X$,LEN(X$)-A)
210 '
220 '** LINKS  AFKAPPEN **
230 '
240 DEFFN L$(X$,A)=RIGHT$(X$,LEN(X$)-A)
250 '
260 '** GEDEELTE WEGHALEN **
270 '
280 DEFFN M$(X$,S,L)=LEFT$(X$,S-1)+
                FNL$(X$,S+L-1)
290 '
300 '** CENTREREN        **
```

```
310 '
320 DEFFN C$(X$,L)=RIGHT$(SPACE$(40)
                 ,L/2-LEN(X$)/2)+X$
330 '
340 '** SCORE-FORMAAT   **
350 '
360 DEFFN SC$(S,L)=RIGHT$(STRING$(L,"0")
                 ,L-LEN(STR$(S))+1)+
                 RIGHT$(STR$(S),LEN(
                 STR$(S))-1)
370 '
380 '** DEMONSTRATIE **
390 '
400 SCREEN 0:WIDTH 39:KEY OFF
410 D$="demonstratie"
420 FOR G=1 TO LEN(D$)
430  PRINT FN R$(D$,G);" ";FN L$(D$,G)
440 NEXT
450 :
460 LOCATE 0,15
470 PRINT D$
480 PRINT FN M$(D$,3,5)
490 :
500 T$="DIT:"          :PRINTFN C$(T$,39)
510 T$="IS EEN"        :PRINTFN C$(T$,39)
520 T$="GECENTREERDE":PRINTFN C$(T$,39)
530 T$="TEXT"          :PRINTFN C$(T$,39)
540 :
550 :
560 SR=100
570 FOR G=1 TO 10
580  LOCATE 30,G
590  PRINT FN SC$(SR,6)
600  SR=SR+150
610 NEXT
620 :
```

```
630 LOCATE 0,22
640 A$=INPUT$(1)
650 CLS:KEY ON:END
```

High/low funkties

Twee funkties die de high- en de low-byte van een getal berekenen.

FN H(x) ; geeft de high-byte van x
FN L(X) ; geeft de low-byte van x

```
100 '============================
110 '
120 '   NIEUWE FUNCTIES VOOR
130 '
140 '     LOW EN HIGH BYTE
150 '
160 '============================
170 '
180 '** HIGH-BYTE **
190 '
200 DEFFN H(X)=INT(X/256)
210 '
220 '** LOW -BYTE **
230 '
240 DEFFN L(X)=X-FN H(X)*256
250 '
260 '** DEMONSTRATIE **
270 '
280 CLS:KEY OFF
290 LINEINPUT"GEEF GETAL :";G$
300 G=VAL(G$)
310 LOCATE 2,5
320  PRINT "LOW  BYTE =";
330  PRINT FN L(G)
340 LOCATE 2,7
350  PRINT "HIGH BYTE =";
360  PRINT FN H(G)
370 A$=INPUT$(1)
380 GOTO 280
```

Deek-funktie

De deek-funktie voert een double-byte-peek uit. Dat wil zeggen hij leest twee opeenvolgende lokaties uit en berekent aan de hand van de inhoud het overeenkomstige getal.

FN DEEK(x) ; geeft inhoud van x & x+1

```
100 '============================
110 '
120 '     NIEUWE FUNCTIE:
130 '
140 '      DEEK( adres )
150 '
160 '============================
170 '
180 :
190 DEFFN DEEK(X)=PEEK(X)+256*PEEK(X+1)
200 :
210 '** DEMONSTRATIE **
220 :
230 CLS
231 :
232 HIMEM =&HFC4A
233 BOTTM =&HFC48
234 :
240 PRINT"TOP VAN BASIC  :";
250 PRINT FN DEEK( HIMEM )
255 :
260 PRINT"START VAN BASIC:";
270 PRINT FN DEEK( BOTTM )
275 :
280 PRINT:END
```

USR adres uitlezen

Machinetaalroutines moet u over het algemeen met een USR-call aanroepen. Vaak is het startadres van zo'n USR-routine niet direkt te vinden, zodat het moeilijk wordt om een routine te bekijken en/of aan te passen.

Met de volgende formule kunt u van iedere USR-call het startadres uitlezen. De formule luidt:

```
PEEK(x*2+&HF39A) + PEEK(x*2+&HF39B)*256
```

waarbij 'x' staat voor de USR-call (0-9) als funktie:

```
DEF FNU(x)=PEEK(x*2+&HF39A)+PEEK(x*2+&HF39B)*256
```

of bijvoorbeeld:

```
DEF FN U$(x)=HEX$(PEEK(x*2+&HF39A)+PEEK(x*2+&HF
39B)*256)
```

MSX grafisch

3D sinusgrafiek

Programma's om funkties in drie dimensies weer te geven (althans de suggestie van 3 dimensies te wekken) zijn er al verscheidene geschreven. Met de meest ingewikkelde berekeningen en na een hééééle lange tijd verschijnt dan toch, puntje na puntje, een prachtige 3-D grafiek.

Maar waarom zoveel moeite? Dat men op een veel eenvoudigere manier de suggestie van 3-dimensies kan opwekken bewijst het onderstaande programma. Het geeft dan wel niet zo'n fraai resultaat als de eerstgenoemde programma's, maar het resultaat mag er best zijn.

Naar believen kunt u de funktie-parameters in regel 140 en de stepgrootte in regel 130 veranderen om wat te experimenteren. Om de snelheid hoeft u het niet te laten; in luttele sekonden is het scherm opgebouwd.

```
10 '============================='
20 '                             '
30 '    DIT PROGRAMMA TEKENT     '
40 '                             '
50 '     EEN 3-DIMENSIONALE      '
60 '                             '
70 '        SINUS-CURVE          '
80 '                             '
90 '============================='
100 COLOR 1,15,15
110 KEYOFF
115 '
120 SCREEN 2
130 FOR X=20 TO 255 STEP 2
140    YY=SIN(X/10)*50
150    LINE(X-20,0)-(X,90-YY)
160    LINE        -(X-20,190)
170 NEXT X
175 '
180 A$=INPUT$(1)
190 END
```

Variatie 1

Dit programma is een variatie op de vorige. De funktie is veranderd zodat een veel indrukwekkender beeld ontstaat met meer dieptewerking.

Ook hier geldt: experimenteren staat vrij om nog leukere beelden te krijgen.

```
10 '==============================`
20 '                              `
30 '         VARIATIE 1           `
40 '                              `
50 '       ` GEBERGTE '           `
60 '                              `
90 '==============================`
100 COLOR 1,15,15
110 KEYOFF
115 '
120 SCREEN 2
130 FOR X=0 TO 255 STEP 2
140    YY=(COS(X/5)+2*SIN(X/20))*20
150    LINE(X-20,0)-(X,90-YY)
160    LINE          -(X-20,190)
170 NEXT X
175 '
180 A$=INPUT$(1)
190 END
```

Variatie 2

Het bijzondere aan deze variatie is dat het op SCREEN3 werkt. Het effekt komt ongeveer overeen met de eerste routine, maar er moet eigenlijk nog wel wat aan geknutseld worden om de dieptewerking te versterken. Iets voor u?

```
10 '==============================`
20 '                              `
30 '        VARIATIE 2            `
40 '                              `
50 '        3D-SINUS              `
60 '                              `
70 '      OP  SCREEN-3            `
80 '                              `
90 '==============================`
100 COLOR 1,15,15
110 KEYOFF
115 '
120 SCREEN 3
121 '
125 C(1)=9:C(2)=8:C(3)=6 :' KLEUR
126 '
127 C=0
130 FOR X=0 TO 215 STEP 4
140   YY=SIN(X/20)*50
145   C=C+1:IF C=4 THEN C=1
150   LINE(X+40,0)-(X,90-YY) ,C(C)
160   LINE         -(X,190   ),C(4-C)
170 NEXT X
175 '
180 A$=INPUT$(1)
190 END
```

Circle met variabele stapgrootte

Het CIRCLE-kommando in MSX-Basic heeft één nadeel; het is altijd een cirkel uit één lijn. Je kunt de afstand van de verschillende puntjes op de cirkel niet variëren.

Onderstaand programma tekent een cirkel aan de hand van cosinus en sinus en heeft als voordeel dat je nu wél de stapgrootte in kunt geven.

De routine die de cirkel tekent wordt aangeroepen met 'GOSUB 1000'. Van te voren dient u wel een aantal variabelen in te stellen.

XC	=	X-koördinaat middelpunt cirkel
YC	=	Y-koördinaat middelpunt cirkel
RA	=	Radius of straal van de cirkel
SG	=	Stapgrootte tussen twee puntjes op de cirkel

Het nadeel van deze routine is dat het een stuk langzamer gaat dan de standaard CIRCLE routine. Maar wellicht kan het u toch van dienst zijn.

```
10 '==============================`
20 '                              `
30 '   DIT PROGRAMMA TEKENT       `
40 '      EEN CIRCEL MET          `
50 '  VARIABELE STAPGROOTTE.      `
60 '                              `
70 ' XC =>  X-COORDINAAT          `
72 ' YC =>  Y-COORDINAAT          `
74 ' RA =>  RADIUS (STRAAL)       `
76 ' SG =>  STAP GROOTTE          `
78 '                              `
80 '       GOSUB 1000             `
85 '                              `
90 '==============================`
100 COLOR 1,15,15
```

```
110 SCREEN 2
115 '
120 XC=100:YC=100:RA=50:SG=10
130 GOSUB 1000
135 '
140 XC=50:YC=120:RA=30:SG=5
150 GOSUB 1000
155 '
160 A$=INPUT$(1)
170 END
997 '
998 '== TEKEN CIRCEL ==
999 '
1000 PI=4*ATN(1):SG=SG*PI/180
1010 FOR P=0 TO 2*PI STEP SG
1020   X=RA*COS(P)
1030   Y=RA*SIN(P)
1040   X=.75*X+XC:Y=Y+YC
1050   PSET(X,Y)
1060 NEXT P
1070 RETURN
```

ROM pokes

De routines in dit hoofdstuk hebben allemaal betrekking op het programma "ROM wordt RAM", en werken alleen wanneer van te voren "ROM wordt RAM" een keer gerund is.

De pokes veranderen de ROM niet echt, dus u hoeft niet bang te zijn dat u uw computer onherstelbaar zou beschadigen. Zo gauw als u een RESET geeft of de computer uit en aanzet, is alles weer volkomen normaal.

Hoogte kursor instellen

Normaal gesproken is de kursor 8 pixels hoog en bedekt daardoor een heel karakter. In de INSERT mode wordt die hoogte gehalveerd. Door in de betreffende ROM-routines deze hoogte te veranderen, kunnen leuke effekten bereikt worden.

De lokaties waar het om gaat zijn de volgende:

&H0A0A = de hoogte van de normale kursor (8)
&H0A12 = de hoogte van de insert kursor (3)

Bij PC's wordt de kursor vaak als een klein, dun streepje onder de letters weergegeven. Om op uw MSX dit effekt te bereiken, typt u

```
POKE &H0A0A,1
```

Om de boel eens flink te neppen, zou u het volgende kunnen doen:

```
POKE &H0A0A,3:POKE &H0A12,8
```

Drukt u eens op de INSERT-toets en kijk wat er bebeurt.

Dansende kursor

Deze routine verandert bij iedere toetsdruk de hoogte van de kursor. Dit grappige effekt laat zich goed illustreren door op de spatiebalk te drukken en deze ingedrukt te houden.

De routine verlaagt de hoogte van de kursor (adres &HA0A) totdat deze 1 is. Vervolgens wordt deze weer verhoogd totdat het maximum (=8) bereikt is waarna het proces weer opnieuw begint.

```
1 '================================
2 '        DANSENDE CURSOR
3 '
4 ' DEZE ROUTINE LAAT DE CURSOR
5 '
6 '          SWINGEN !!!
7 '
8 '( HET ROM MOET IN RAM STAAN)
9 '================================
10 '
15 CLEAR200,(PEEK(&HFC4A)+
           256*PEEK(&HFC4B))-&H2B
20 '
23 A=PEEK(&H15C):A=A+1:POKE&H15C,A
24  IF PEEK(&H15C)<>A THEN CLS:PRINT
      "EERST`ROM WORDT RAM'RUNNEN":END
25 POKE &H15C,A-1
27 '
30 SA=PEEK(&HFC4A)+256*PEEK(&HFC4B)
35 SO=SA
41 DEF FNH(X)=INT(X/256)
42 DEF FNL(X)=X-FNH(X)*256
100 '
110 '
120 READ H$:IF H$="*" THEN 200
130  FOR H=1 TO LEN(H$) STEP 2
```

```
140    C$=MID$(H$,H,2)
150    IF C$=" A" THEN 180
151    P=VAL("&H"+C$)
160    POKE SA,P
170    SA=SA+1
180   NEXT H
190 GOTO 120
200 '
210 '======= AKTIVEREN =======
220 '
230 '= HOOK
240 '
250 POKE &HFDCD,FNL(SO)
255 POKE &HFDCE,FNH(SO)
260 POKE &HFDCC,&HCD
261 '
262 '= FLAG
265 '
266 POKE &HFF2E,0
270 END
999 '
1000 DATA F5     :' PUSH AF
1010 DATA 3A2EFF:' LD   A,(FLAG)
1020 DATA B7     :' OR   A
1030 DATA 2012   :' JR   NZ,OMHOOG
1040 DATA 3A0A0A:' LD   A,(CURSH)
1050 DATA 3D     :' DEC  A
1060 DATA 320A0A:' LD   (CURSH),A
1070 DATA FE01   :' CP   1
1080 DATA 2017   :' JR   NZ,END
1090 DATA 3E01   :' LD   A,1
1100 DATA 322EFF:' LD   (FLAG),A
1110 DATA 1810   :' JR   END
1114 '
1115 '               ** OMHOOG **
1116 '
```

```
1120 DATA 3A0A0A:' LD   A,(CURSH)
1130 DATA 3C    :' INC  A
1140 DATA 320A0A:' LD   (CURSH),A
1150 DATA FE08  :' CP   8
1160 DATA 2005  :' JR   NZ,END
1170 DATA 3E00  :' LD   A,0
1180 DATA 322EFF:' LD   (FLAG),A
1184 '
1185 '               ** END **
1186 '
1190 DATA F1    :' POP  AF
1200 DATA C9    :' RET
1210 DATA *
```

Tab-kode veranderen

Je zou denken dat een tab-toets erg gemakkelijk is. Met een paar toetsdrukken een heel eind verder op het scherm, net zoals dat bij een typmachine het geval is over het papier.

Niets is echter minder waar; de MSX tab-toets heeft de vervelende eigenschap alles wat hij tegenkomt in spaties te veranderen. Gaat u maar eens met de kursor op het begin van een programmaregel staan en druk dan maar eens op de tab-toets. Alles verdwenen.

Dankzij het "ROM wordt RAM" programma kunnen we dit eens gaan verbeteren. De kode die de tab-toets produceert, staat op adres:

&H2406 (normaal: 32 = spatie)

In ons geval zouden we de volgende POKE moeten ingeven:

POKE &H2406,28

Kode 28 betekent *kursor naar rechts*, dus precies wat we wilden. Nu rest ons echter nog een probleem; ook het BASIC-kommando tab(x) geeft spaties. De kode voor dit kommando staat op adres

&H4AF6 (normaal: 32 = spatie)

Dus de oplossing luidt:

POKE &H4AF6,28

Natuurlijk kunt u ook andere kodes op de betreffende lokaties poken. Experimenteren is één van de leukste dingen wat POKE's betreft.

Restore met variabele

Het RESTORE kommando werkt op twee manieren: zonder regelnummer en met een regelnummer. Dit regelnummer *moet* echter een konstante, bijvoorbeeld 1500 zijn en geen variabele. Dit in tegenstelling tot andere Basic-dialekten waarbij het soms wel mag.

Door in de ROM de RESTORE routine te veranderen, kunnen we dit aanpassen.

De betreffende POKES zijn:

```
POKE &H63D1,&H2F
POKE &H63D2,&H54
```

Nu kan men bijvoorbeeld in een programma de volgende regel ingeven:

```
100 a=2500
110 RESTORE a
...
```

Een nuttige toepassing voor deze uitbreiding is bijvoorbeeld een *adventure*, waar het handig kan zijn een restore op een berekend regelnummer uit te voeren.

Tips voor machinetaal

Waarden doorspelen via USR

Veel beginnende machinetaalprogrammeurs gebruiken het USR-statement alleen om hun routines te starten. Vaak weten ze wel dat het op één of andere manier mogelijk moet zijn om een getal mee te geven aan het USR-statement en terug, maar hoe het precies in zijn werk gaat is vaak niet duidelijk. Niet in de laatste plaats komt dit door de ronduit slechte dokumentatie die er in de beschikbare boeken over te vinden is.
Meestal wordt de lezer overspoeld met termen als 'single' en 'double' precision, 'mantisse', 'bcd' en ga zo maar door. Voor de gemiddelde routineschrijver is dit allemaal niet belangrijk. Hij/zij wil gewoon een getal of variabele doorspelen om overbodige POKEn te voorkomen. Hoogst eenvoudig.

Een voorbeeld:

stel we willen een getal opsplitsen in LOW en HIGH byte. Laten we dan twee USR-routines schrijven:

USR0(getal)	geeft de LOW byte
USR1(getal)	geeft de HIGH byte

In een Basic-programma zouden we dan moeten kunnen schrijven:

```
100 G=5000
110 L=USR0(G) : H=USR1(G)
```

Wat gebeurt er nu als we een kommando als USR1(G) uitvoeren?
Twee dingen:

- de machinetaal-routine waar USR1(G) voor staat wordt aangeroepen,
- bij het begin van die routine staat op lokatie &HF7F8 en &HF7F9 respektievelijk de LOW en de HIGH byte van de inhoud van variabele G. Is G bij aanroep bijvoorbeeld gelijk aan 257 dan staat op lokatie &HF7F8 het getal 2 en op &HF7F9 het getal 1 (1*256+2=257);

Dit getal kunnen we uitleze met de instruktie

```
LD HL,(F7F8H)
```

Willen we nu een getal teruggeven aan Basic, dan doen we in feite precies hetzelfde.

Stel we willen de inhoud van HL terug geven aan Basic. De volgende drie regels zijn dan nodig:

```
LD   (F7F8H),HL    ;HL wordt op &HF7F8 en &HF7F9
                    gezet
LD   A,2           ; A laden met TYPE
LD   (F663H),A     ; TYPE op &HF663 zetten
RET                ; terug naar Basic
```

Het TYPE dient om aan te geven dat het gaat om een getal dat slechts uit LOW en HIGH byte bestaat (INTEGER heet dat). Als u dit laatste (nog) niet geod begrijpt, ligt u er dan maar niet wakker van. Als u de routine zoals die hierboven staat overneemt in uw programma's kan er niets verkeerd gaan.

Terug naar ons voorbeeld. Dit is nu niet zo moeilijk meer;

```
USR0   LD   HL,(F7F8H)    ; HL=waarde tussen haakjes doorge-
                            geven
       LD   H,0           ; alleen de LOW-byte blijft over
       LD   (F7F8H),HL    ; teruggeven aan Basic
       LD   A,2
       LD   (F663H),A     ; TYPE
       RET                ; terug naar Basic
```

```
USR1   LD   HL,(F7F8H)    ; HL=waarde tussen haakjes doorge-
                            geven
       LD   A,H           ; HIGH-byte bewaren
       LD   HL,0          ; HL op nul zetten
       LD   L,A           ; alleen de HIGH-byte staat nu in HL
       LD   (F7F8H),HL    ; teruggeven aan Basic
       LD   A,2
       LD   (F663H),A     ; TYPE
       RET                ; terug naar Basic
```

Ik hoop dat het aan de hand van deze voorbeelden duidelijk is geworden. Veel oefenen en uitproberen is de beste manier om het onder de knie te krijgen.

Vrije geheugenruimte

Vaak is het probleem bij een machinetaalroutine: wáár in het geheugen plaats ik die routine om niet in konflikt te komen met andere programma's en routines. Hier volgen enkele tips om dit probleem op te lossen.

Voor routines die minder dan 24 bytes lang zijn kunt u een stukje RAM gebruiken dat (op MSX1 in ieder geval) niet gebruikt wordt. Deze ruimte loopt van &HFC82 tot &HFC9A.

Van &HFFC6 tot &HFFFE zijn 56 bytesruimtes voor kleine routines. (Ook hier geldt: voor MSX1 is dit zeker; voor MSX2 waarschijnlijk ook maar ik durf dat niet voor honderd procent te garanderen.)

Indien u in uw programma géén PLAY gebruikt, dan staat er een groot blok van 384 bytes ter beschikking en wel van &HF975 tot &HFA75. (Het gebruik van SOUND heeft hier geen invloed op.)

Verder zijn er nog ongebruikte lokaties in het RAM die u als (tijdelijke) opslag kunt gebruiken. Deze lokaties zijn:

&HF69D	2 bytes
&HF6B3	2 bytes
&HF7BC	8 bytes
&HF857	8 bytes
&HF866	22 bytes
&HFCAD	1 byte
&HFD09	128 bytes

Stop op deze lokaties geen blijvende informatie of hele routines maar slechts tijdelijke waarden die niet opnieuw nodig zijn.

Index
van alle truuks en tips delen

Hoofdstuk **deel/pag.**

Nederlandstalige MSX boeken

MSX Basic handboek voor iedereen – A.C.J. Groeneveld
Hèt nederlandstalige handboek voor iedere MSX gebruiker. Omvat een volledige behandeling van het MSX Basic, alles uiteraard in helder nederlands. Het handboek geeft een antwoord op iedere vraag die een programmeur over MSX Basic zou kunnen stellen. De laatste mogelijk nog aanwezige onduidelijkheden worden weggenomen door de zinvolle voorbeelden. ISBN 90 6398 100 7

MSX Zakboekje – Wessel Akkermans
Een naslagwerk voor ná of naast het Handboek. Onder andere: nietcomputergerichte tabellen, de MSX Basic instruktieset, Z80 instruktieset, hardwaregegevens (konnektoren) en een aantal programmaatjes.
ISBN 90 6398 888 5

MSX Disk handboek – A.C.J. Groeneveld
Voor diskdrivebezitters om naast het Handboek te gebruiken. Behandeling van het disk-gebeuren en de specifieke disk kommando's, uitgebreid met voorbeelden, tabellen en overzichten, tekentafelprogramma en een basisprogramma voor basisonderhoud. ISBN 90 6398 407 3

MSX Quick disk handboek – A.C.J. Groeneveld
Hèt handboek voor iedere quick disk gebruiker. ISBN 90 6398 254 2

MSX DOS handboek – A.C.J. Groeneveld
Geeft een totaalbeeld van de mogelijkheden met MSX DOS; met een uitgebreide inleiding op het begrip 'operating system'.
ISBN 90 6398 674 2

MSX Machinetaalhandboek – Hans Klopper/Marcel le Belle
De grondbeginselen van machinetaal kennen, is de ideale manier om werkelijk alles uit de MSX computer te halen. Daarom dit boek. Enkele van de behandelde onderwerpen: scroll routine, machinetaal op kassette schrijven, disassembler, lijst van ROM-routines, alle hookadressen... Hèt handboek voor iedere MSX programmeur die zijn computer ten volle wil benutten. ISBN 90 6398 735 8

MSX2 Basic handboek – A.C.J. Groeneveld
Alles over MSX2 Basic, de grafische- en geluidsmogelijkheden en de computer zelf in 507 pagina's. Met 288 voorbeeldprogramma's.
ISBN 90 6398 221 6

MSX2 Disk/Dos uitbreidingshandboek – A.C.J. Groeneveld
De volledige behandeling van het MSX2 Disk Basic en het MSX DOS operating system, met vooraf een zeer duidelijke inleiding tot de feno-

menen disk en operating system. Verder praktische tabellen, duidelijke afbeeldingen en zinvolle voorbeelden. ISBN 90 6398 222 4

MSX2 Utility-toepassingshandboek – A.C.J. Groeneveld
Een verzameling programma's die voor elke MSX-er onontbeerlijk zijn. Mogelijkheden met deze programma's: bestandsonderhoud met lijstwerk, diagrammen maken, binair manipuleren binnen blokken op schijf. Alle programma's voor MSX en MSX2. ISBN 90 6398 223 2

MSX en MSX2 mogelijkheden – Wessel Akkermans
Wat kan- wat mankeert- wat kan ik met mijn computer. Welk type MSX heb ik? Doen mijn joysticks het wel goed? Werken alle toetsen naar behoren, maakt mijn schijveneenheid lees- of schrijffouten? Op al deze vragen krijgt u antwoord bij het uitvoeren van het in dit boek beschreven programma. De beschreven programmatuur is leerzaam voor elke MSX-er en bovendien een waardevol gereedschap bij het onderzoeken van de goede werking van MSX en MSX2 computers voor alle gebruikers. ISBN 90 6398 606 8

MSX2 Machinetaalhandboek – Hans Klopper/Marcel le Belle
De ideale manier om alles uit uw computer te halen, is kennis te nemen van de machinetaal. De machinetaalfunkties worden op een heldere manier uitgelegd en vaak ondersteund door bijpassende voorbeelden. Een greep uit de inhoud: scroll-routine, disassembler, kassette-diskettekonversie, alle hook-adressen, Basic tokens, systeemvariabelen, bespreking van de MSX2 video chip, overzicht Z80 instrukties.
ISBN 90 6398 915 6

MSX2 Zakboekje – Wessel Akkermans
In dit ene boek zijn zoveel mogelijk nuttige gegevens voor de MSX en MSX2 programmeur samengebracht, waar mogelijk in de vorm van overzichten en tabellen. Om u het zoeken in ver

MSX Basic voor kinderen – H.C. de Heer
In twee delen worden kinderen bekend gemaakt met de MSX computer en het programmeren daarvan. Aan de hand van programma's die in de loop van de boekjes worden opgebouwd, wordt een frisse start gemaakt voor echt programmeren. Voor kinderen vanaf 8 jaar.
ISBN 90 6398 084 1/ISBN 90 6398 304 2

MSX truuks en tips
Als MSX-er moet je wel even wat tijd uittrekken om het MSX Basic te leren kennen, maar beheers je het eenmaal, dan kun je met een paar instrukties de meest ongelofelijke dingen doen. Programmeren is en blijft echter een kunst. In het op een korte en krachtige wijze oplossen van problemen herkent men de ware programmeur. En achter alle

boekjes van de serie MSX truuks en tips gaan zulke ware programmeurs schuil. Boordevol waardevolle truuks, tips, routines en programma's, in elk geval altijd razend interessant.
deel 1: ISBN 90 6398 900 8, 2: 90 6398 340 9, 3: 90 6398 910 5,
deel 4: ISBN 90 6398 897 4, 5: 90 6398 745 5, 6: 90 6398 879 6,
deel 7: ISBN 90 6398 789 7, 8: 90 6398 850 8

MSX verder uitgediept – H. Klopper
Alle belangrijke RAM en VRAM adressen, uitleg van de video chip en zijn registers, diskloader utility en een uiterst geavanceerde programmabeveiliging, kassette-disk-konversie en een aantal interessante programma's. ISBN 90 6398 447 2

MSX Basic met vpoke en sprite toepassingen – J.G. Ottenhoff
Wie zegt, dat u niet kunt programmeren? – De ideale kursus om Basic kennis toe te passen; op een populaire manier en met een knipoog gebracht. ISBN 90 6398 372 7

Computer en modemgebruik – Wessel Akkermans/Piet den Heijer
Alle gegevens die nodig zijn om suksesvol te kunnen deelnemen aan het gegevensverkeer tussen uw computer en die van anderen.
Voor alle homecomputers ISBN 90 6398 798 6
Voor personal computers ISBN 90 6398 070 1

MSX en datakommunikatie – Christian Rakow
Alles over Videotex en Bulletin Board Systems; de wegwijzer voor iedere modemgebruiker. ISBN 90 6398 959 8

MSX leerboeken – Wessel Akkermans en Piet den Heijer
De serie MSX leerboeken geeft een komplete kursus MSX Basic programmeren in drie delen. En voor wie alles van het MSX2 Basic wil weten, is er een vierde deel dat alle extra's van MSX2 behandelt. De gebruikte voorbeelden stellen de lezer in staat om al in een vroeg stadium bruikbare programma's te maken. Aan het eind van ieder deel is een voorbeeldprogramma opgenomen, dat de lezer/leerling laat zien waartoe hij na bestudering van het betreffende leerboek in staat zal zijn. De bijbehorende opdrachtenboekjes met uitwerkingen maken de serie zelfs uitstekend geschikt voor het onderwijs.
Basic leerboek deel 1 ISBN 90 6398 649 1 - Opdrachten 90 6398 596 7
Basic leerboek deel 2 ISBN 90 6398 769 2 - Opdrachten 90 6398 556 8
DOS leerboek deel 3 ISBN 90 6398 519 3 - Opdrachten 90 6398 516 9
MSX2 leerboek deel 4 ISBN 90 6398 737 4

MSX Software Plus

MSX Introtape – A.C.J. Groeneveld
Deze kassette introduceert MSX op een uiterst vrindelijke en onder-

wijzende manier. Na het starten volgt een demonstratie van de mogelijkheden van MSX. Tenslotte twee 'les' gedeelten. In anderhalf tot drie uur weet u wat de MSX computer is, wat hij kan, en heeft u haast ongemerkt al wat regels geprogrammeerd. ISBN 90 6398 148 1

MSX Script – Ton Weijters
Volledig menugestuurde nederlandstalige tekstverwerker, geschikt om efficiënt teksten te bewerken. Korrigeren, zoeken, blokken verplaatsen, kopiëren of verwijderen, onderstrepen, vet zetten en volledige pagina-opmaak behoren tot de mogelijkheden van deze tekstverwerker. ISBN 90 6398 189 9

MSX Draws – A.C.J. Groeneveld
Een tekenprogramma waarmee u al binnen 10 minuten uw eerste tekening kunt maken. Werkt vriendelijk, gebruikmakend van alle grafische mogelijkheden van de MSX computer. Effektief bereik: ruim 30.000 bij 30.000 puntjes met mogelijkheden als cirkels, krommen, verkleinen, verschuiven, verdraaien en substitueren. ISBN 90 6398 754 4

truuks en tips
deel 8

Hoe meer truuks en tips deeltjes er verschijnen, hoe moeilijker het wordt om met nieuwe, originele routines te komen.
Toch blijkt MSX echter zo flexibel en doordacht te zijn, dat het nog wel een paar jaartjes zal duren voordat iemand hem helemaal van top tot teen kent. Zo zal er voorlopig nog genoeg kennis over te dragen zijn om nog heel wat boekwerken te vullen. Dit truuks en tips deeltje is daar een voorbeeld van.
Het menu is rijk gevarieerd, zodat er voor elk wat wils is. Bijvoorbeeld de antiscroll-routine, waarmee we de computer foppen met 25 regels op het scherm; een sprite editor voor kassettegebruikers, werken met joystick èn toetsenbord, 3D sinusgrafiek genereren, zelf instellen van de kursorhoogte. En eindelijk een volledig foutloze versie van ''Cload en runnen''.
Verder een tweetal wat langere routines, waarvan velen veel plezier zullen hebben. De eerste is een ''Smooth scroll'' programma, dat teksten vloeiend over het scherm laat scrollen. De tweede is de topper in dit boek. Het is een utility waarmee *meer* dan 80 tekens per regel op het scherm kunnen worden geschreven.
Bepaal nu zelf de repetitiesnelheid van de toetsen, maak een screendump door de tab-toets aan te raken, genereer nederlandse foutmeldingen *zonder* geheugenruimteverlies...

Eindeloze mogelijkheden met MSX, en eindeloze mogelijkheden met MSX truuks en tips. Dit deel bewijst dat zelfs een achtste deel uitdagend en fascinerend kan zijn.

MSX truuks en tips. Voor elke MSX-er een bron van lering en vermaak. Bij ieder deel opnieuw.

ISBN 90 6398 850 8 uitgeverij Stark-Texel b.v.